基于软件体系结构开发的案例

llliao@seu.edu.cn

票务系统架构设计案例分析

- 1 项目背景
- 2 需求分析
- 3 系统架构设计
- 4 小结

1. 项目背景

由于票务种类的繁多，客户信息的量大复杂。所以在其管理上存在较大困难，特别是早期单用人力和纸张进行管理。导致信息的不全面和错误率高，加之存储介质的约束，难以长期有效的管理。

随着计算机网络的发展，电子商务的普及。**一种基于B/S模式的票务系统**提出了需求。由于票务的特殊性，需要**系统有很强的稳定性，要求较快的反应速度，响应多点同时请求**。另外**后台对票务的所有相关信息需要完全记录。完成历史信息的保存，查询；对当前信息的录入，查询，修改，删除**。

2. 需求分析

主要任务

创建代表“目前”业务情况的业务模型，并将此业务模型转换成“将来”的系统模型，包括功能需求和非功能需求。非功能需求又包括质量属性和各种约定。

通过对客户的当前业务的分析，我们得到当前业务的基本需求。

功能需求

功能	说明
客户信息管理	用户的创建、登录、删除和维护
票务信息管理	票务的添加、删除和维护
票务查询	查看相应的票务信息
预定购票	票务的预定、购买和取消

非功能需求

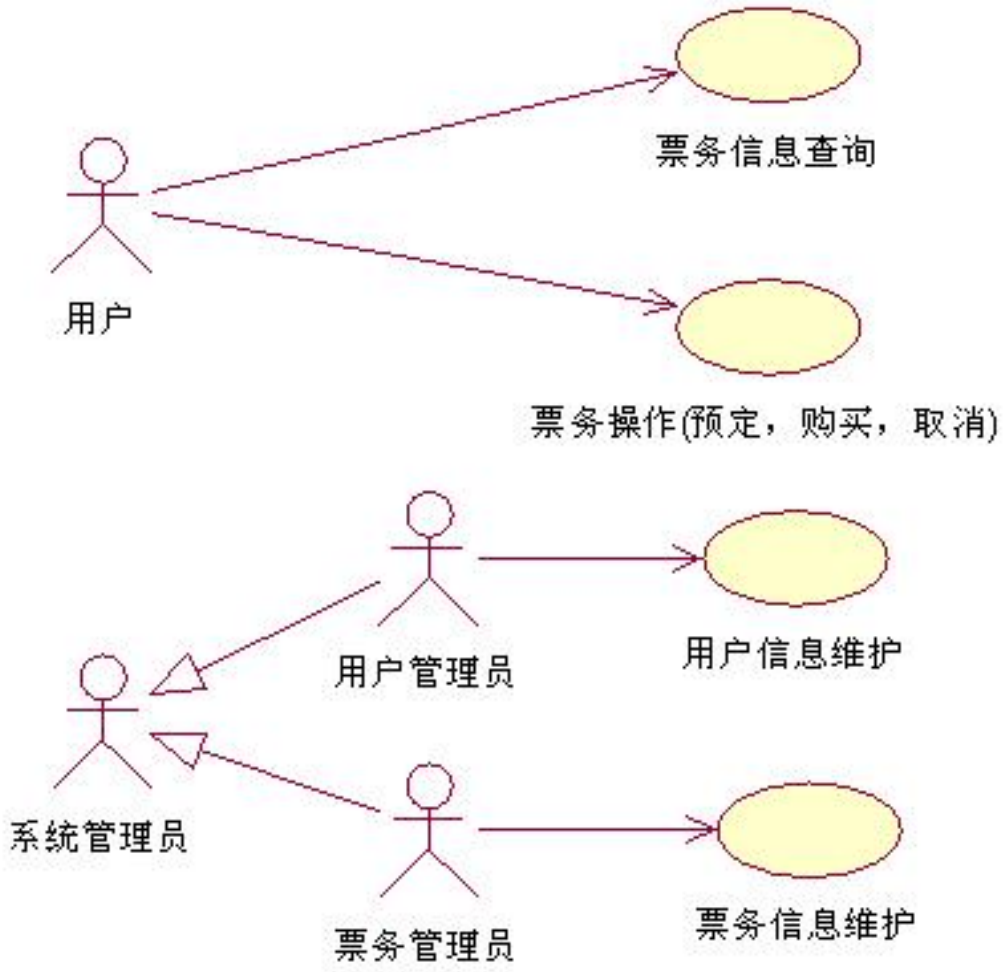
质量属性	说明
性能	用户访问的系统应该能在规定的时间内做出响应，如果系统由于网络或者数据库原因不能在规定时间内做出反应，那么系统应该提出警告，不能出现用户无故长时间等待的情况。
安全性	在web数据库客户端，web服务器和数据库服务器之间，都应该有防火墙保护，防止网络上的非法数据请求。
易用性	不同的用户应该能够以不同形式访问不同的内容
可用性	系统提供7X24小时的服务，且很少停机
可测试性	系统的各部分易于单独测试，并能方便地进行整体测试

2.1 定义系统

根据业务的功能需求，该系统主要的涉众有**系统管理人员和客户**，系统管理人员又分为**票务管理人员**和**用户管理人员**。票务管理人员会对票务信息进行相关维护，用户管理人员对客户信息进行相关的维护。由此得出系统角色，分析其对系统的具体要求，并找出系统的各个用例。

用例	说明
票务信息查询	用户输入相关查询条件信息，查看到相关票务的具体信息，当查询条件不符合规定时，系统给出相应提示。
票务操作	用户根据查询出来的票务信息对票务信息进行预订，购买，取消等操作
票务信息维护	票务管理员对票务信息进行维护，如添加，删除等
用户信息维护	用户管理员根据用户资料，维护系统中记录的用户相关信息。
……	……

根据上述分析，可以得到系统用例视图：



2.2 细化定义

(1) 细化用例

细化业务用例模型，是为了更加详细地分析和描述用例。同时，将业务用例模型转换成系统的用例模型。下面，以“用户” 角色进行票务购买为例。

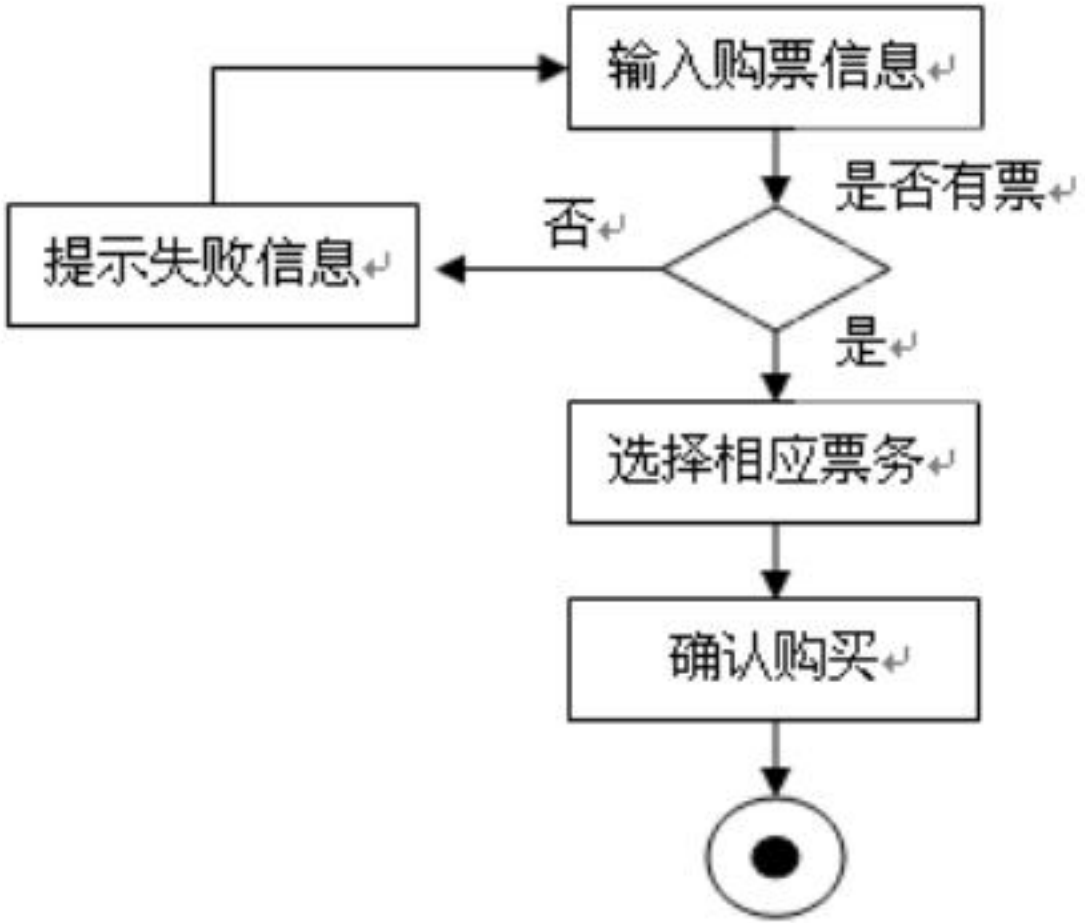
细化用例后，还需对用例进行详细描述，直到所有涉众都认可描述的内容已经能够正确表达出他们的需求为止。在RUP方法论中指明通过阐述一个用例的名称、简要描述、事件流、特殊需求、前置条件和后置条件等六个方面可以对用例进行描述。下面以用例“用户购买票务”为例细化描述。

要素	说明
用例名称	用户购买票务
简要描述	用户根据当前票务信息购买相应票务
事件流	基本事件流 （1）用户在购票的名称栏中输入要购买的票务的起始地与目的地 （2）系统根据客户输入，列出相应的票务信息 （3）用户根据自己的实际情况选择符合自己相应条件的票务，如票价、时间等。 （4）系统显示购买成功，或者显示交易失败。 （5）该“用户购买票务”用例结束。

“用户购买票务”用例细化描述（续）

要素	说明
备选事件流	系统查询不到票务相关信息，则按下面步骤进行： （1）提示用户票务交易无法进行，并给出交易失败原因 （2）其次，撤销此次交易的记录。
特殊需求	系统不可伪造数据，交易失败原因要合理并且详尽
前置条件	用户必须先登陆
后置条件	交易成功后数据库及时更新票务信息

上面对用例的描述仅限于文字描述，还不够形象。再以活动图的形式进行建模描述如下：



(2) 结构化用例

结构化用例的目的是通过观察这些已经细化的用例，看能不能抽取出共有的、可选的行为，把这些共同的内容建立为新的用例。这样的好处是可以消除冗余的需要，以及改善系统整体需求内容的可维护性。像“用户信息维护”用例中，“查询用户信息”应作为一个新的用例提取出来，以提高上面所说的需求内容的可维护性。

即：ABSD方法的步骤 中的此步

（1）架构需求——功能分解

一个设计元素有一组功能，这些功能必须分组。分解的目的是使每个组在体系结构内代表独立的元素。分解可以进一步细化。

功能的分组可选择几个标准：

1）功能聚合。

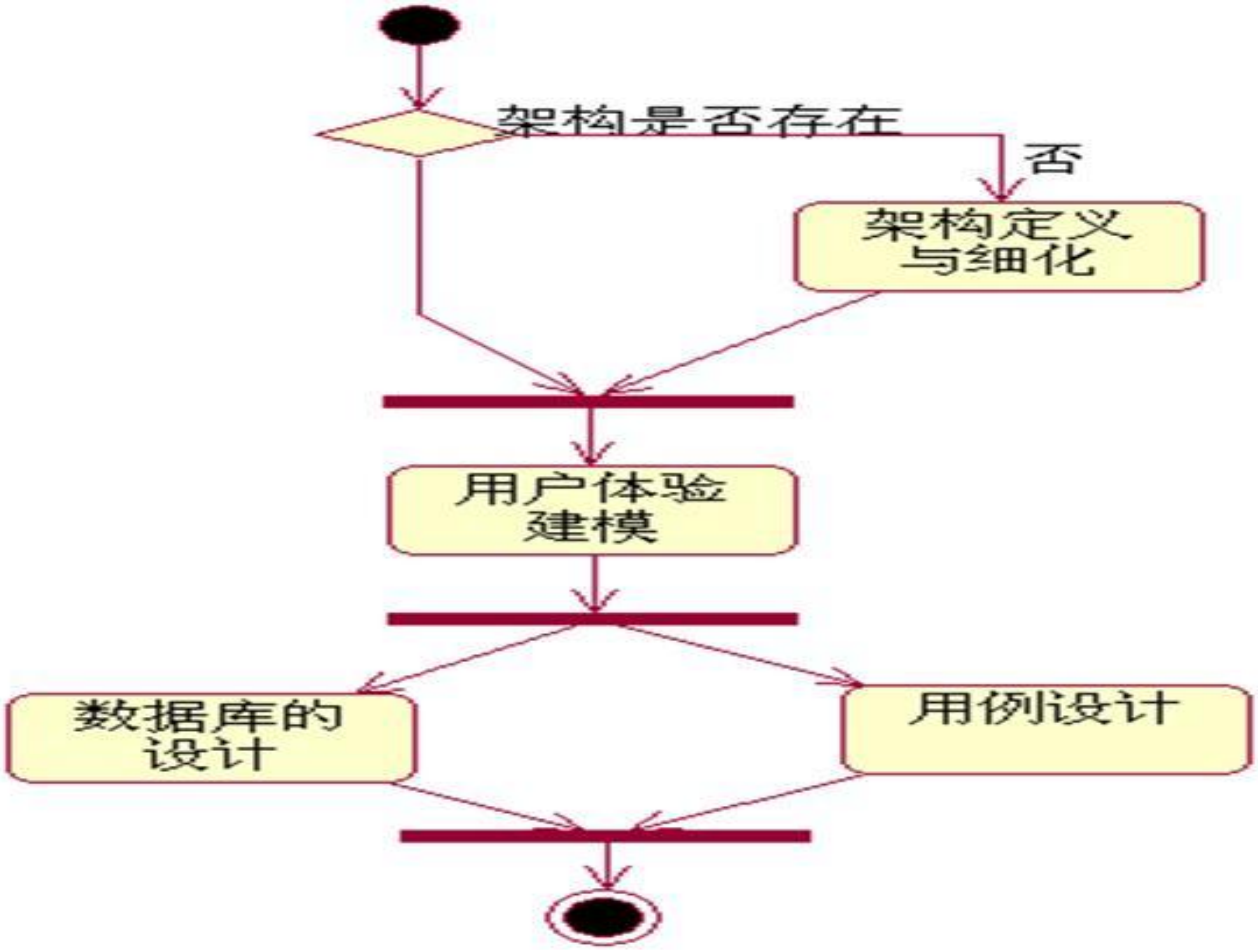
2）数据或计算行为的类似模式。

3）类似的抽象级别。

4）功能的局部性。公共功能区分出来。

3. 系统架构设计

将需求内容转换成设计模型的雏形以及用户体验模型，其目的是建立整个系统初步的解决方案，为详细设计活动打下基础，这一阶段的具体活动如下：

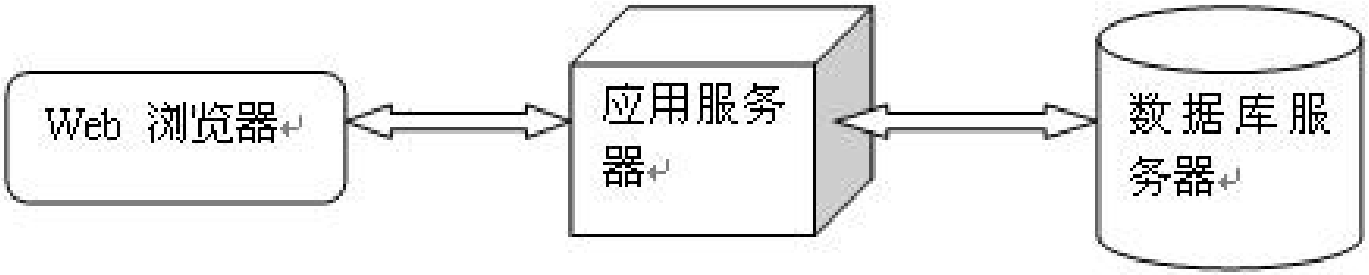


3.1 体系结构的选择

早期的票务系统仅仅针对售票单位，只是简单的数量控制，票务记录。而新的票务系统不仅仅具有以前的所有功能，还利用网络将客户包括进来。方便客户进行操作，利用网络可以让客户在任何有网络的地方就可以直接连入系统。又由于计算机的支持，数据库中有所有客户的信息，可以方便售票方对客户进行管理，提供更好的服务。

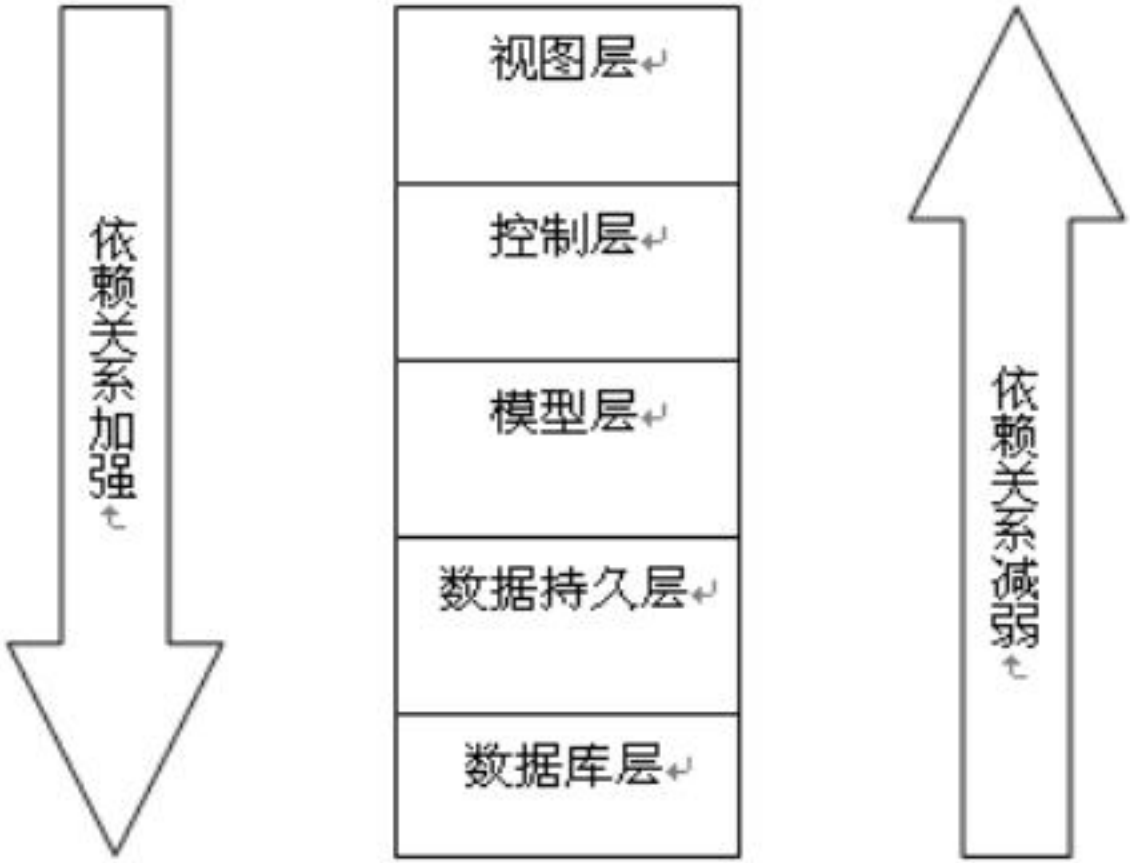
本系统采用基于B/S的分层结构。这种结构有如下特点：节省投资、跨地域广；维护和升级方式简单，如果想对功能修改，可以方便的进行更改，大大减少维护成本。

系统的结构视图如下：



- 在J2EE开发中，搭配良好的框架可以降低开发人员解决复杂问题的难度，而如何将框架整合起来，以使每一层都向另外的层次以松散的方式来提供接口，同时让组合的三个框架在每一层都以一种松耦合的方式彼此沟通，从而与低层的技术透明无关，这就是框架分析的目的和要求。
- 根据前期对需求的分析，决定采用基于SSH框架来构建此分布式的信息管理系统。
- SSH是 struts+spring+hibernate的一个集成框架，是目前（当年）比较流行的一种Web应用程序开源框架。
- SSH把Structs、Hibernate和Spring组合起来，目标就是希望能实现系统的“低耦合、高内聚”。也就是要求系统易于维护、易于适应变更、可重用性的特点。

SSH多层的构架模式，从上到下依次为视图层、控制器层、模型层、持久化层和数据库层，如下图所示：



视图层：职责是提供控制器，将页面的请求委派给其他层进行处理，为显示提供业务数据模型。

控制层：职责是按预定的业务逻辑处理视图层提交的请求。（1）处理业务逻辑和业务校验

（2）事务管理

（3）管理业务层对象间的依赖关系

（4）向表示层提供具体业务服务的实现类

模型层：职责是将模型的状态转交视图层或者控制层，以提供页面给浏览器。

数据持久层：职责是建立持久化类及其属性与数据库中表及其字段的对应关系。提供简化SQL语句的机制。实现基本的数据操作（增、删、读、改）

数据库层：数据库的建立与管理。

规则（约束）：

（1）系统各层次及层内部子层次之间都不得跨层调用。

（2） 由bean传递模型状态。

（3）需要在表示层绑定到列表的数据采用基于关系的数据集传递。

（4）对于每一个数据库表（Table）都有一个DB Entity class与之对应，由Hibernate完成映射。

（5）有些跨数据库或跨表的操作（如复杂的联合查询）也需要由Hibernate来提供支持。

（6） 表示层和控制层禁止出现任何SQL语句。

SSH框架介绍：

视图层、控制器层用Structs框架来实现，模型层的功能Spring来完成。持久化层时使用Hibernate实现，在这层使用了DAO模式。

Structs应用MVC模型使页面展现与业务逻辑分离，做到了页面展现与业务逻辑的低耦合。当充当表示层的页面需要变更时，只需要修改该具体的页面，不影响业务逻辑Bean等；同样，业务逻辑需要变更的时候，只需要修改相应的Java程序。

使用Spring能运用IoC技术来降低了业务逻辑中各个类的相互依赖：假如，类A因为需要功能F而调用类B，在通常的情况下类A需要引用类B，因而类A就依赖于类B了，也就是说当类B不存在的时候类A就无法使用了。使用了IoC，类A调用的仅仅是实现了功能F的接口的某个类，这个类可能是类B，也可能是类C，由Spring的XML配置文件来决定。这样，类A就不再依赖与类B，耦合度降低，重用性得以提高。

使用Hibernate能让业务逻辑与数据持久化分离，就是将数据存储到数据库的操作分离。在业务逻辑中只需要将数据放到值对象中，然后交给Hibernate，或者从Hibernate那里得到值对象。至于用DB2、Oracle、MySQL还是SQL Server，如何执行的操作，与具体的系统无关，只需在Hibernate的相关的XML文件里根据具体系统配置好。

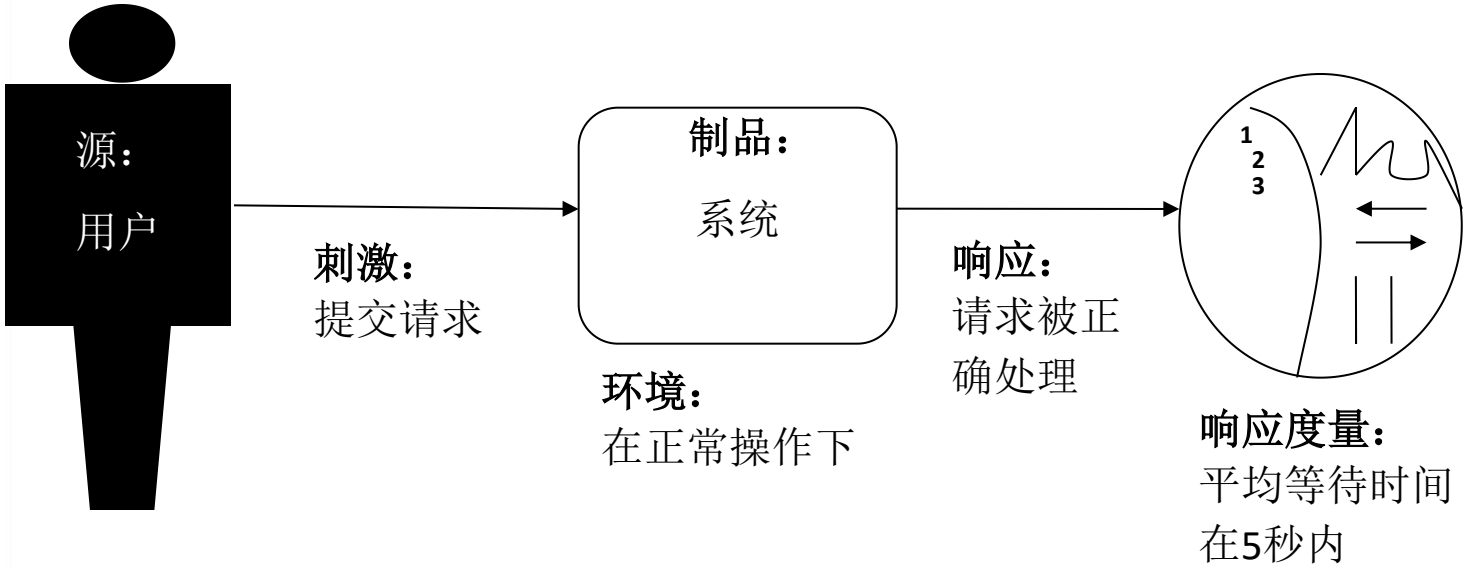
3.2 系统架构的分析与设计

3.2.1 质量属性

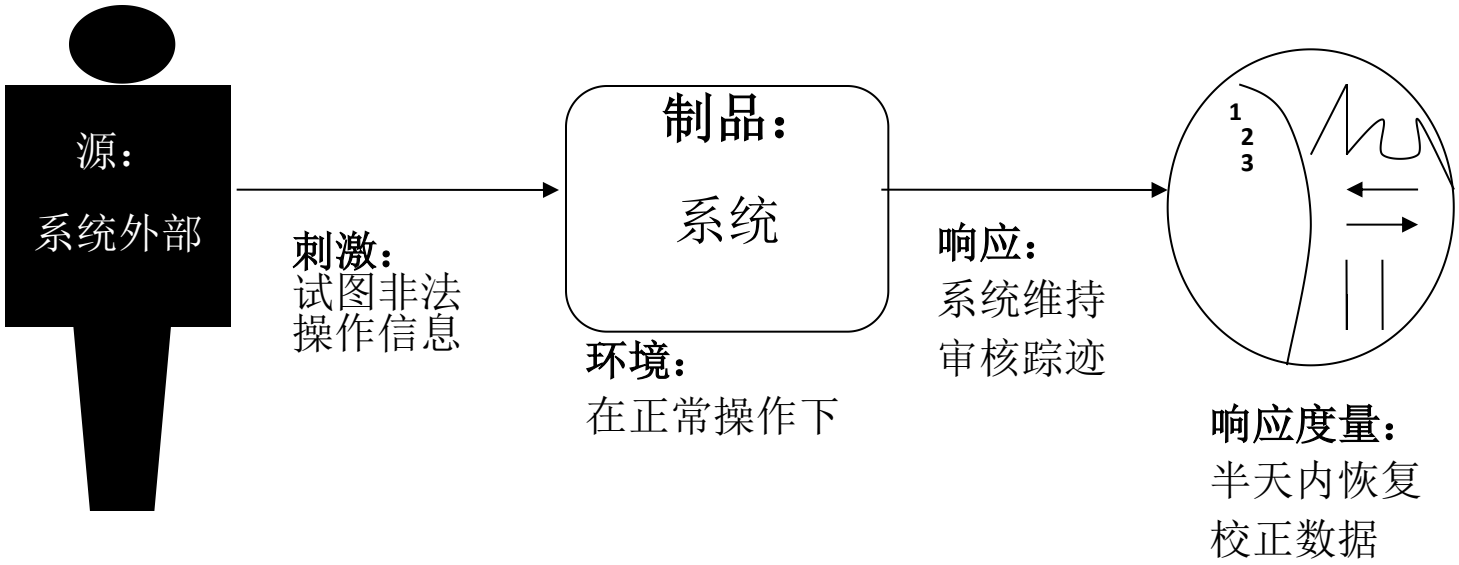
现在我们通过下表来看看构架是如何来满足系统的关键质量属性需求。

目标	实现方式	所采用的战术
性能	用户访问的系统应该能在规定的时间内做出响应，如果系统由于网络或者数据库原因不能在规定时间内做出反应，那么系统应该提出警告，不能出现用户无故长时间等待的情况。	限制队列大小 缓冲池技术
易用性	遵从 J2EE 的系统提供了诸如 JSP 和 servlet 这样的 Java 技术，它们支持内容的渲染，以满足不同用户的需要。用户对系统的操作能得到正确及时的反馈。	单独的用户接口 支持用户主动
安全性	遵从 J2EE 的系统提供了由容器进行授权校验的基于角色的安全性机制，以及已经为使用做好准备的在程序中进行授权检查的安全性机制，在多个用户进行并发操作时保持数据的一致性，有效性。	身份验证 授权 数据机密性 验证码 维护完整性
可用性	当系统试图超出限制范围来进行票务查询或者订购票时必须进行错误检测并且抛出异常，中止进一步的错误操作。遵从 J2EE 的系统提供了可以使用的事务服务，通过提供内建的故障恢复机制，提高了应用的可用性和可靠性	异常检测 内建故障恢复机制 资源调度策略

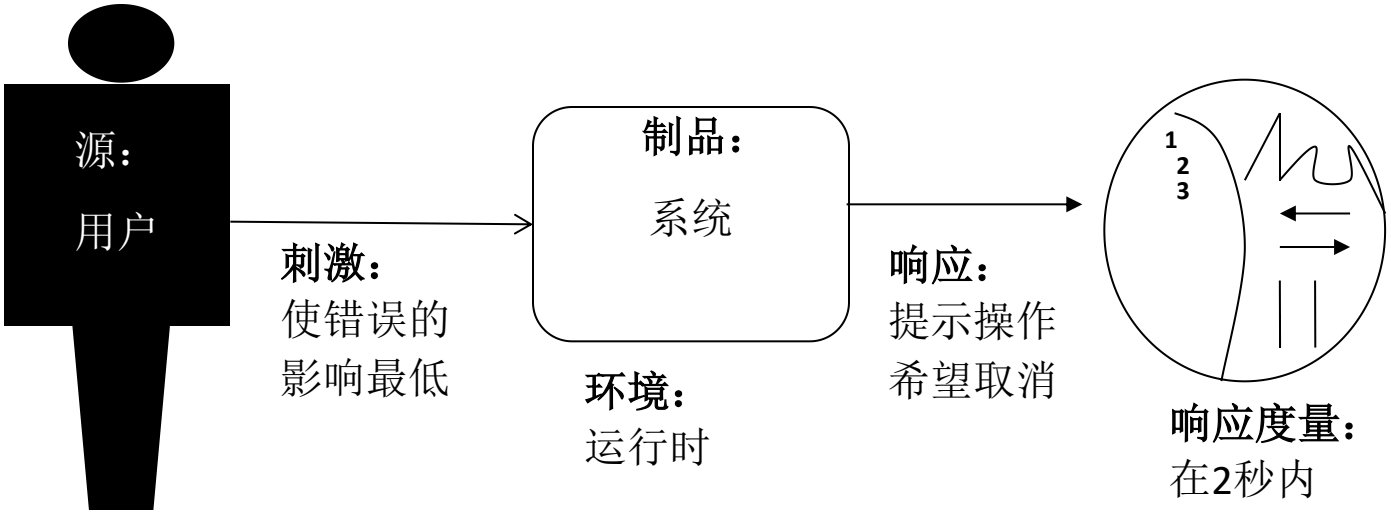
1）性能场景：在系统处于高峰时期，保证登陆的每个顾客所作的选择和查询的响应时间能在5s以内，如果需要等待则给出有友好的提示。系统可以保证以最快速度同时响应500个用户的操作。



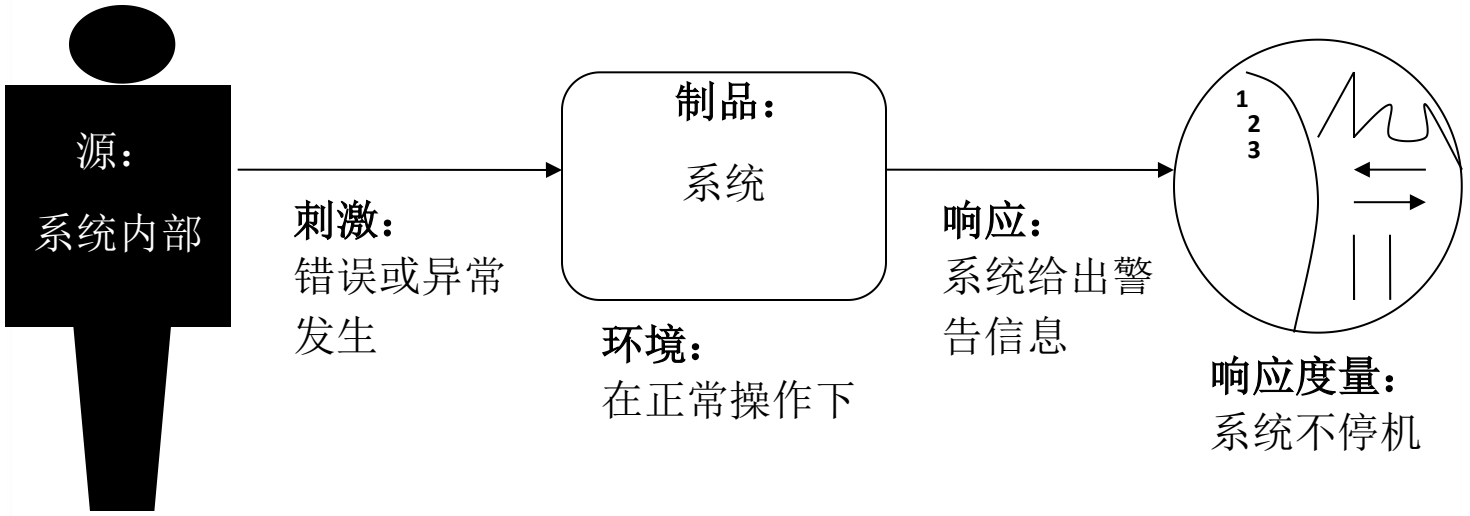
2）安全性场景：杜绝非法用户试图绕过应用服务器直接连接到数据库服务器的端口上，防止非法窃取注册用户个人息；屏蔽某IP短时间内的大量无意义的访问，以防被挤爆，使正常用户无法使用。保证系统数据的机密性和完整性。



3）易用性场景：在该系统中，用户希望在运行时能尽快取消某操作使错误的影响降到最低，取消在1秒内发生；要求具有基本电脑操作常识的人，可以根据良好的界面设计迅速学会使用方法，让熟手用户使用快捷键。



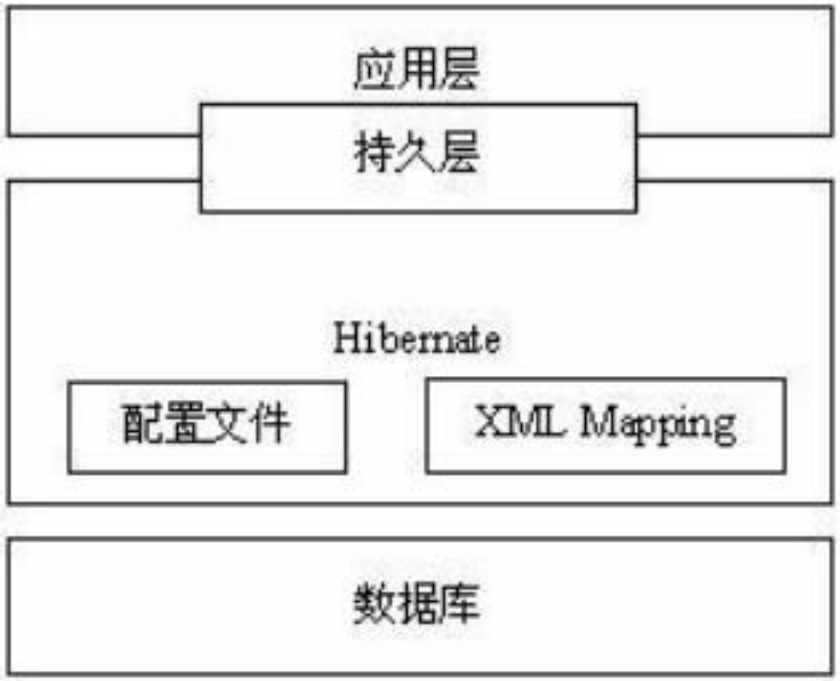
4) 可用性场景：在正常的工作时间内，系统必须具有极高的可用性，保证出故障几率最低。出现故障时系统有相应的处理机制。



3.2.2 数据持久层的架构分析

在数据持久层，我们使用Hibernate来进行处理，通过下图来看看如何通过Hibernate来满足系统的质量属性需求。

Hibernate体系结构概要图



3.2.2 数据持久层的架构分析

Hibernate满足的质量属性需求

目标	实现方式	所采用的办法
性能	当应用程序需要在关联关系间进行导航的时候，由 Hibernate 获取关联对象。同时，Hibernate 的 Session 在事务级别进行持久化数据的缓存操作。	抓取策略 缓存机制
安全性	并发操作时，保证数据的排他性	使用锁机制
易用性	用户在进行 CRUD 操作请求时，可以得到 Hibernate 的及时处理，迅速得到反馈。	封装 JDBC

（1）性能

Hibernate本质上是包装了JDBC来进行数据操作的，由于Hibernate在调用JDBC上面是**优化了JDBC调用**，并且尽可能的使用最优化的，最高效的JDBC调用，所以性能令人满意，同时应用程序需要在关联关系间进行导航的时候，由Hibernate获取关联对象，Hibernate提供的对**持久化数据的缓存机制**也对系统的性能的提高起了很大的作用。

（2）安全性

Hibernate提供的**悲观锁/乐观锁机制**，能够在多个用户进行并发操作时保持数据库中数据的一致性与完整性，避免了对数据库中数据的破坏。

（3）易用性

用户在对票务信息进行操作时都得到Hibernate的支持。

3.2.2 业务逻辑层的架构分析

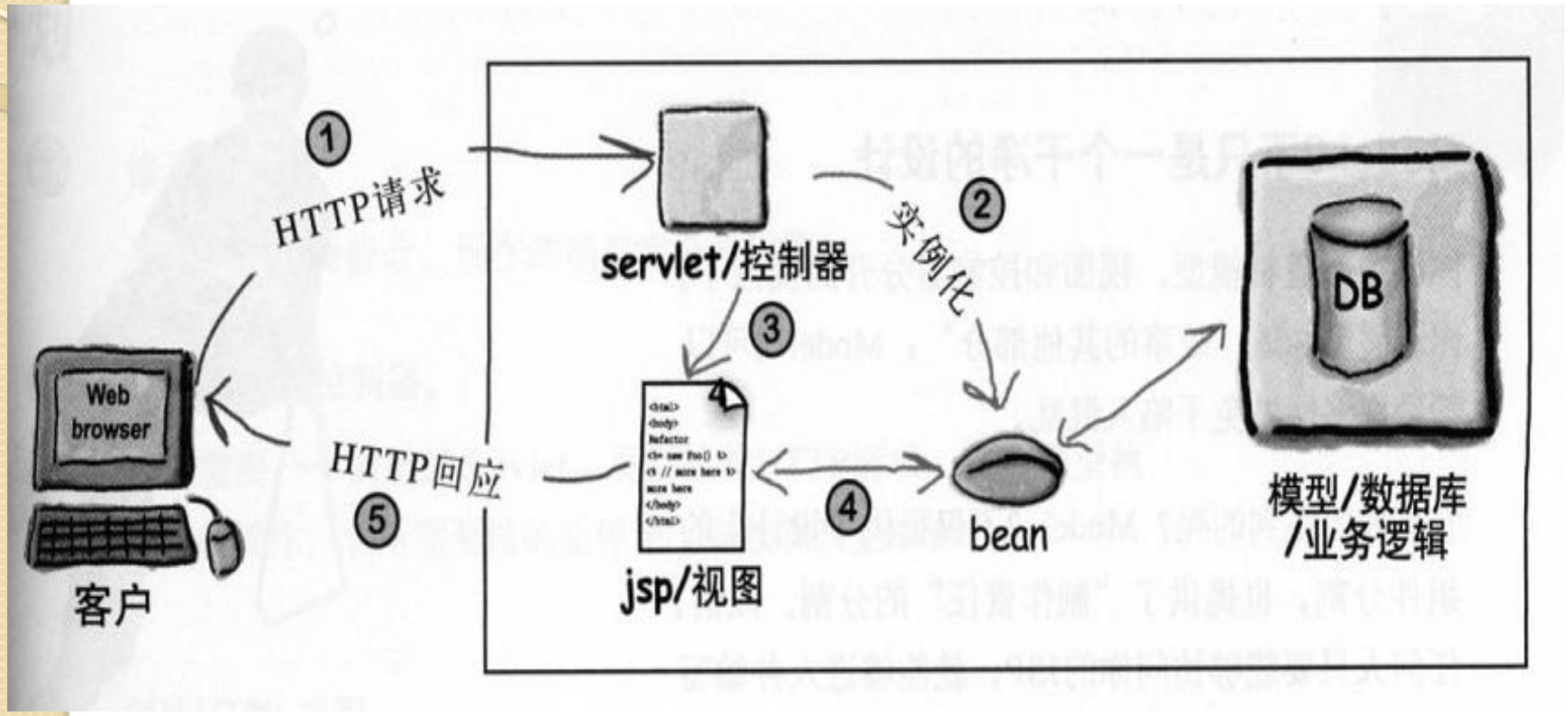
业务逻辑层作为该系统的关键部分，对系统的灵活性实现起着决定性的 作用。在本系统的业务逻辑层架构层中，采取了MVC模式，下面简单介绍一 下MVC模式的好处：

(1) 实现了客户端表示层和业务逻辑层的完全分离

(2) 高效可靠的事务处理

(3) 具有良好的易用性，安全性

MVC模式访问流程：

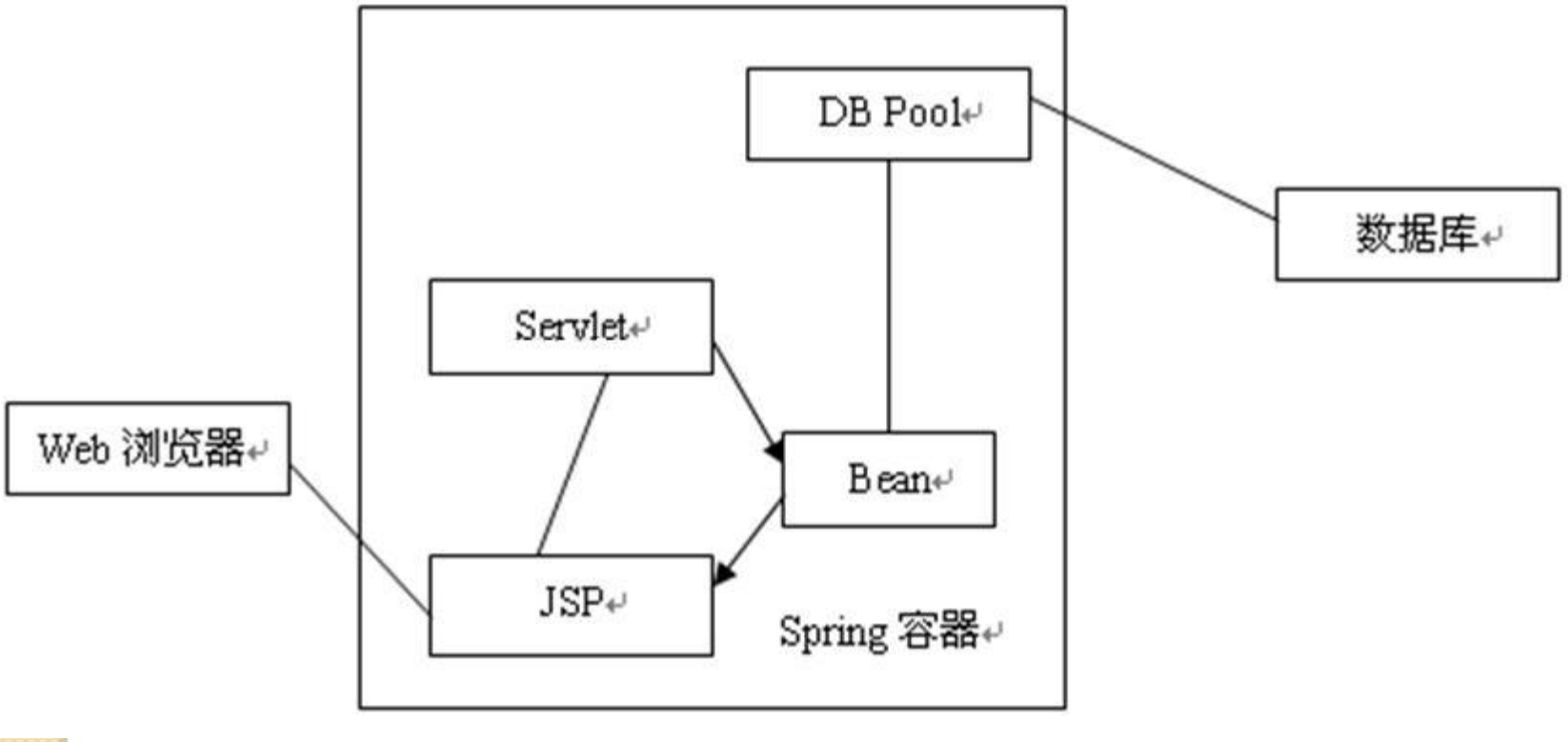

1
HTTP请求
servlet/控制器
2
实例化
3
DB
Web
browser
HTTP回应
5
4
bean
模型/数据库
/业务逻辑
客户
jsp/视图

MVC模式在本系统中应用：

当客户利用网页浏览器，发出HTTP请求时，这通常会牵涉到送出表单数据，例如用户名和密码。Servlet收到这样的数据并解析数据。Servlet扮演控制器的角色，处理你的请求，通常会向模型（一般是数据库）发出请求。处理结果往往以JavaBean的形式打包。视图就是JSP，而JSP唯一的工作就是产生页面，表现模型的视图以及进一步动作所需要的所有控件。当页面返回浏览器作为视图显示出来，用户提出的进一步请求，也会以同样的方式处理。

由于JSP继承了J2EE良好的易用性和安全性，从而为实现系统的关键质量属性奠定了基础。在MVC模式中，视图不再是经典意义上的模型的观察者。当模型发生改变时，视图的确间接的从控制器收到了相当于通知的东西，控制器可以把bean送给视图，以使得视图取得模型的状态。所以，视图在HTTP响应返回到浏览器时只需要一个状态信息的更新。只有当页面被创建和返回时，创建视图并结合模型状态才有意义。这使得提升系统的性能成为可能。只有当相应的操作被执行，系统才会去获取关联对象，并且视图不会直接模型向注册去接受状态信息，使得系统的安全性得到大大提高。

业务逻辑层的框架：



业务逻辑层架构分析：

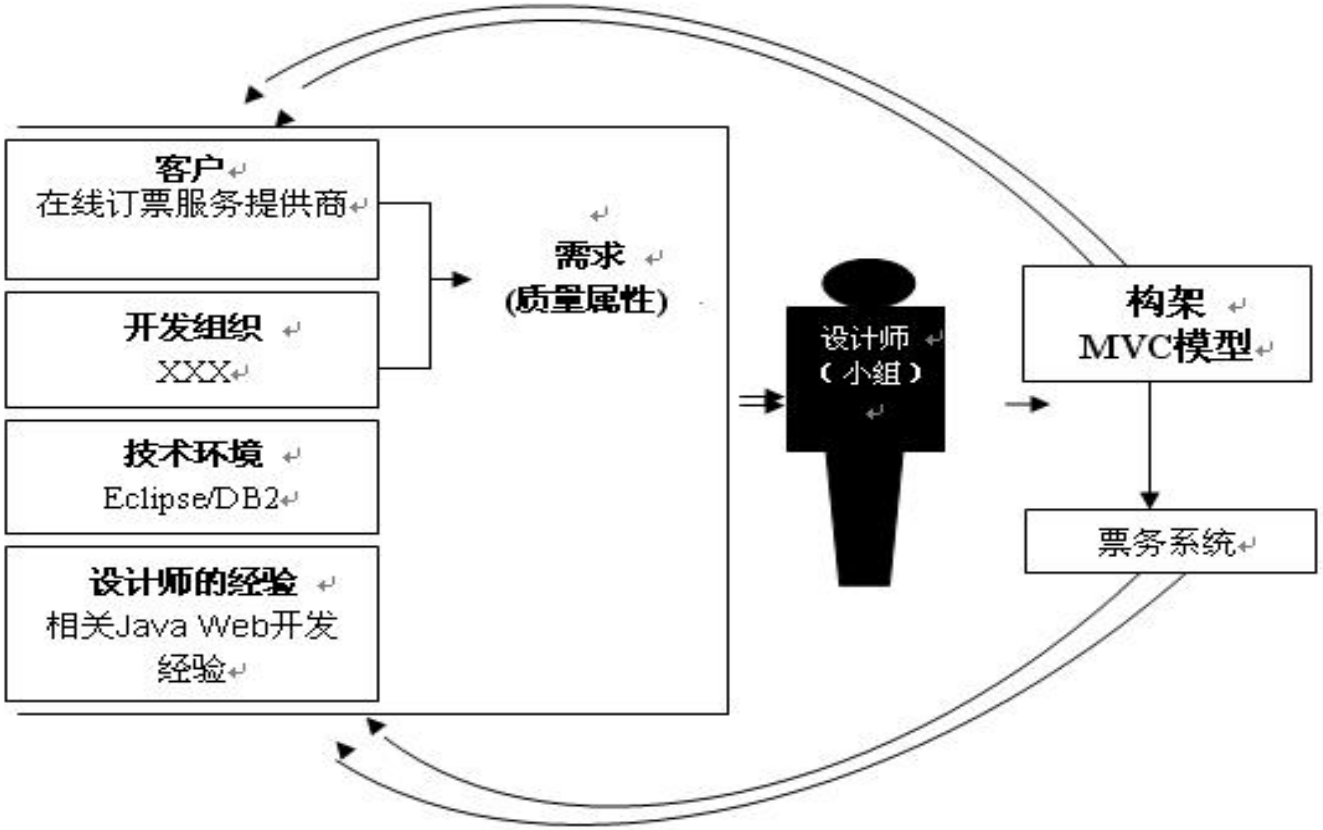
该业务逻辑层的架构是前面MVC模式的一种变形，继承了MVC模式的优点，同时，具体到我们的架构中，它又实现了表示层与业务层的完全分离。在业务逻辑层我们使用Spring框架作为容器，以便实现业务层与表示层和数据层的松耦合。该业务逻辑层架构具备良好的易用性、安全性和性能。

下表给出spring容器如何满足系统关键质量属性

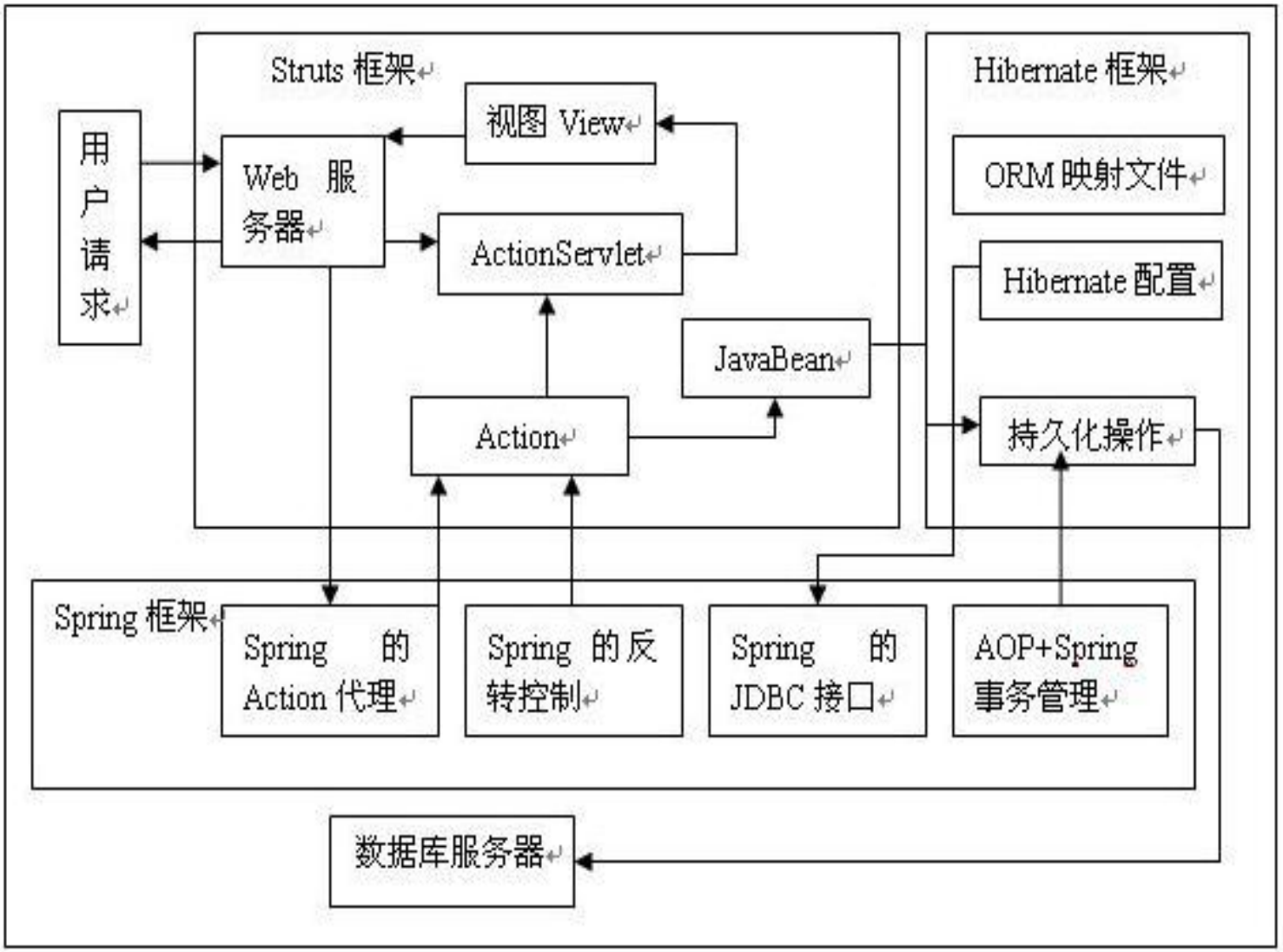
目标	实现方式	所采用的办法
安全性	Spring Framework 利用 AOP 来实现权限拦截，还提供了一个成熟的，简洁清晰的安全框架，通过对 spring bean 的封装机制来实现	AOP Acegi 安全框架
可用性	Spring 框架不再强制开发者在持久层捕捉异常，通常持久层异常被包装成 DataAccessException 异常的子类，将底层数据库异常包装成业务异常，开发者可以自己决定在合适的层处理异常。	异常检测 统一的异常处理机制

3.3 构架表述

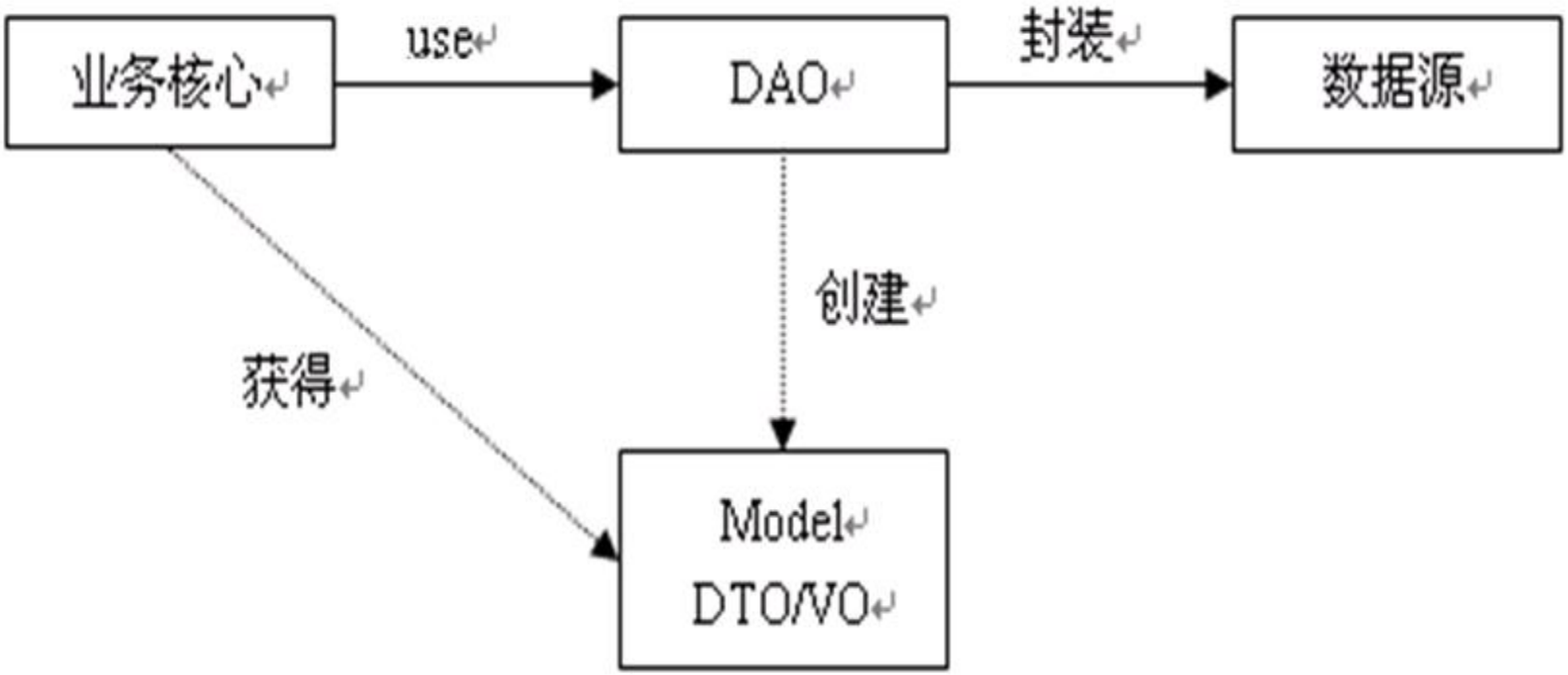
(1) 与构架商业周期的关系



(2) 整体的构架如下：


Struts 框架
视图 View
用户请求
Web 服务器
ActionServlet
JavaBean
Action
Hibernate 框架
ORM 映射文件
Hibernate 配置
持久化操作
Spring 框架
Spring 的 Action 代理
Spring 的反转控制
Spring 的 JDBC 接口
AOP+Spring 事务管理
数据库服务器

因为具体持久化层数据源是多样化的，可能是XML方式或其他不同厂商的关系数据库。我们通过使用DAO模式，业务部分就不用关心具体数据层是如何实现对数据库的操作的，对数据库的操作全部有DAO类实现，如图所示：



科学数据共享网体系结构设计

科学数据共享网


国家科技基础条件平台
NSTI
中国科技资源共享网
English Version
Thursday, October 17, 2013 星期四
用户名
密码
登陆
注册
忘记密码
招聘
首 页 国内资讯 国际资讯 平台简介 地方平台
国际交流 政策法规 建设项目 标准规范 通知通告
仪器设备 自然资源 科学数据 成果转化
科普资源 科学文献 野外观测站
农村医疗服务 气象科学数据 世界科技动态
制造业信息化 小鼠资源库
北京 上海
重庆 广东
大型仪器 自然资源 科学数据 科学文献 科普资源 成果转化 野外观测
请输入关键字
搜索
■ 新疆维吾尔自治区高级信息化管理人才培训班在平台信息技术中心调研学习
■ 首届“共享杯”大学生竞赛
工作动态 >>>
首届“共享杯”大学生科技..
北京市科技信息中心一行参..
山西省阳泉市科技局闫守琦..
科技部人事司、机关党委、..
科技部党组书记、副部长王..
宁波市鄞州区科技信息服务..
2012年度国家科技基础..
湖北省襄阳市别必雄市长一..
科技资讯
国内资讯 >>>
新疆维吾尔自治区高级信息化管理人才培.. 2013-10-17
国内首台500W准连续全固态激光器研.. 2013-10-15
山东省科技成果转化政策效果初显 2013-10-15
2012年全国科技经费投入统计公报 2013-10-15
万钢：数据解说中国科技变化 2013-10-15
科技城科博会正式开展 聚焦“中国创造” 2013-10-15
合肥大型仪器区域中心五年建设获肯定 2013-10-11
浙江推进公共技术资源共享开放实验室帮.. 2013-10-11
中科院在提高玉米耐旱性相关基因研究中.. 2013-10-11
国际首台兼容北斗和GPS掩星探测仪获.. 2013-10-11
国际资讯 >>>
美科学家成功利用诱导多能干细胞制造血管 2013-10-15
新研究给出气候变化进程时间框架 2013-10-15
科学家为近百种“超级增强子”编制目录 2013-10-15
四结光伏电池创转化效率达44.7%.. 2013-10-11
英国研发新药可抑制脑细胞死亡 或有助.. 2013-10-11
三名美国科学家获得2013年诺贝尔化.. 2013
原子厚线型碳线型碳超石墨烯 或成最强.. 2013
印度参与建设世界最大望远镜 2013
世界气象组织：2014年初前出现厄尔.. 2013
2013年诺贝尔物理学奖揭晓 “上帝.. 2013
共享杯大学

科学数据共享网的系统需求

- “中国地球系统科学数据共享网”是国家科学数据共享工程的重要组成部分，同时也是科技部推动“国家科学数据共享工程”2002年试点的三个科学数据共享网之一。
- 该系统针对基于各圈层（大气圈、水圈、 生物圈）相互作用的地球系统科学的整体研究，利用互联网，整合、集成各科研院所、高等院校和国际数据组织以及科学家个人手中的相关专业数据资源，瞄准地球系统科学的前沿研究，开展数据组织、加工与服务，构建物理上分布、逻辑上统一的地球系统科学数据管理与共享服务网。这一工作对于增强我国基础科学研究和前沿科学创新能力具有重要的意义。

数据方面的特殊需求和特点

- 能够快捷地收集数据
  - 科学数据分散在科研院所和科学家手中，要设计开发一套收集数据的机制，使其能够快速地整合到系统中，提供数据共享服务。
  - 数据收集的途径应主要通过网络媒介，而且不能影响系统所提供的网络服务的正常运行。

数据方面的特殊需求和特点

- 有效存储和管理海量数据，并快速定位数据
  - 该系统能够提供目录服务，合理地管理数据。
  - 提供用户查阅、下载、使用数据的服务。
  - 当用户在系统中查找数据时，希望能够快速定位数据，提供服务，平均响应时间最长不超过20秒。

数据方面的特殊需求和特点

- 保护数据版权，保证数据的安全性
  - 科学数据是科学工作者辛勤劳动的果实，同书籍一样也存在版权的问题。所以在数据的使用上，需要版权保护。
  - 此外，由于一些数据有其时效性和保密性，所以在提供服务时需要对数据访问进行相应的安全控制。

系统需求

- 希望“科学数据共享网”能通过Internet为用户提供数据服务，包含：数据目录服务、数据资源导航、数据下载功能、对数据进行稳妥地安全管理。

非功能性需求

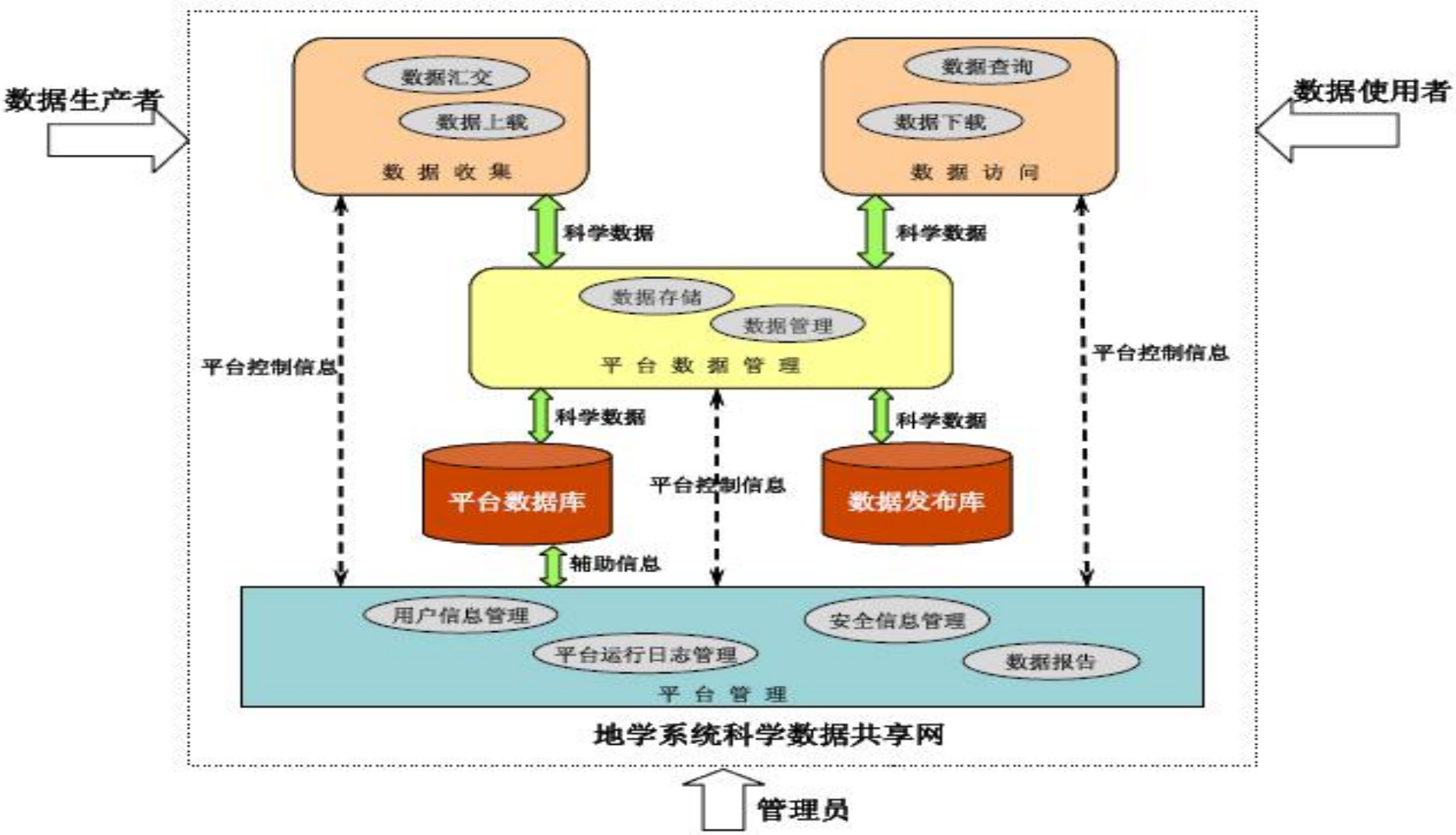
质量属性	针对质量属性的需求
可用性/可靠性	系统应能长期稳定地提供服务，近似7 × 24小时工作强度； 在负载过重或是系统崩溃的情况下，能保证用户的请求不丢失； 当系统出现故障或崩溃时，恢复时间不超过两小时；
可维护性	修改某个子系统或服务时，不影响其他子系统或服务；
性能	高峰时系统的平均响应时间控制在20秒以内； 系统能够满足100个并发的用户查询请求； 系统至少能够支持2000个用户的在线服务；
安全性	对有保密性要求的数据实施安全控制； 提供系统运行日志监控信息，供管理员了解系统的运行和安全状态；
商业属性	2005年中期完成系统，年底前投入正式使用； 能够利用现有系统的可利用资源； 初期总共投资2000万，分别用于系统的集成建设和开发、共享数据标准的制定。

- 科学数据共享网的体系结构？

科学数据共享网的体系结构

- 原型的体系结构及其分析
  - 根据需求，数据将以Internet为传输途径完成共享。在目前以Internet为前提的系统中，应用最广泛的是B/S（Browser /Server）结构。
  - 这样的结构已经相当成熟，并具有很大的灵活性。科学数据共享网也是基于这样初衷而设计的。

系统的原型设计


数据生产者
数据汇交
数据上载
数 据 收 集
数据查询
数据下载
数 据 访 问
数据使用者
科学数据
科学数据
数据存储
数据管理
平 台 数 据 管 理
平台控制信息
平台控制信息
科学数据
科学数据
平台数据库
平台控制信息
数据发布库
辅助信息
用户信息管理
安全信息管理
平台运行日志管理
数据报告
平 台 管 理
地学系统科学数据共享网
管理员

系统的原型设计

- 对于科学数据的存储、管理、共享等诸多计算都是由“中心”服务器承担。在中心服务器中，又划分了数据收集、数据访问、平台数据管理和平台管理四个模块。
  - 数据收集负责收集用户通过Internet上载或是其它途径（光盘、磁盘）提交上来的科学数据。
  - 数据访问负责向用户提供访问科学数据的服务---查询和下载

- 平台数据管理承担了与数据库交互，管理和存储数据的工作。它提供的接口负责将收集的科学数据先暂存在平台数据库中；然后供工作人员对数据进行有效性检查和加工，并将合法数据转移到发布数据库中；最后管理发布数据库中数据的接口提供数据的访问服务。
- 平台管理承担了管理用户信息、管理用户和数据的安全信息，以及生成平台运行日志的任务。

- 是否合适？

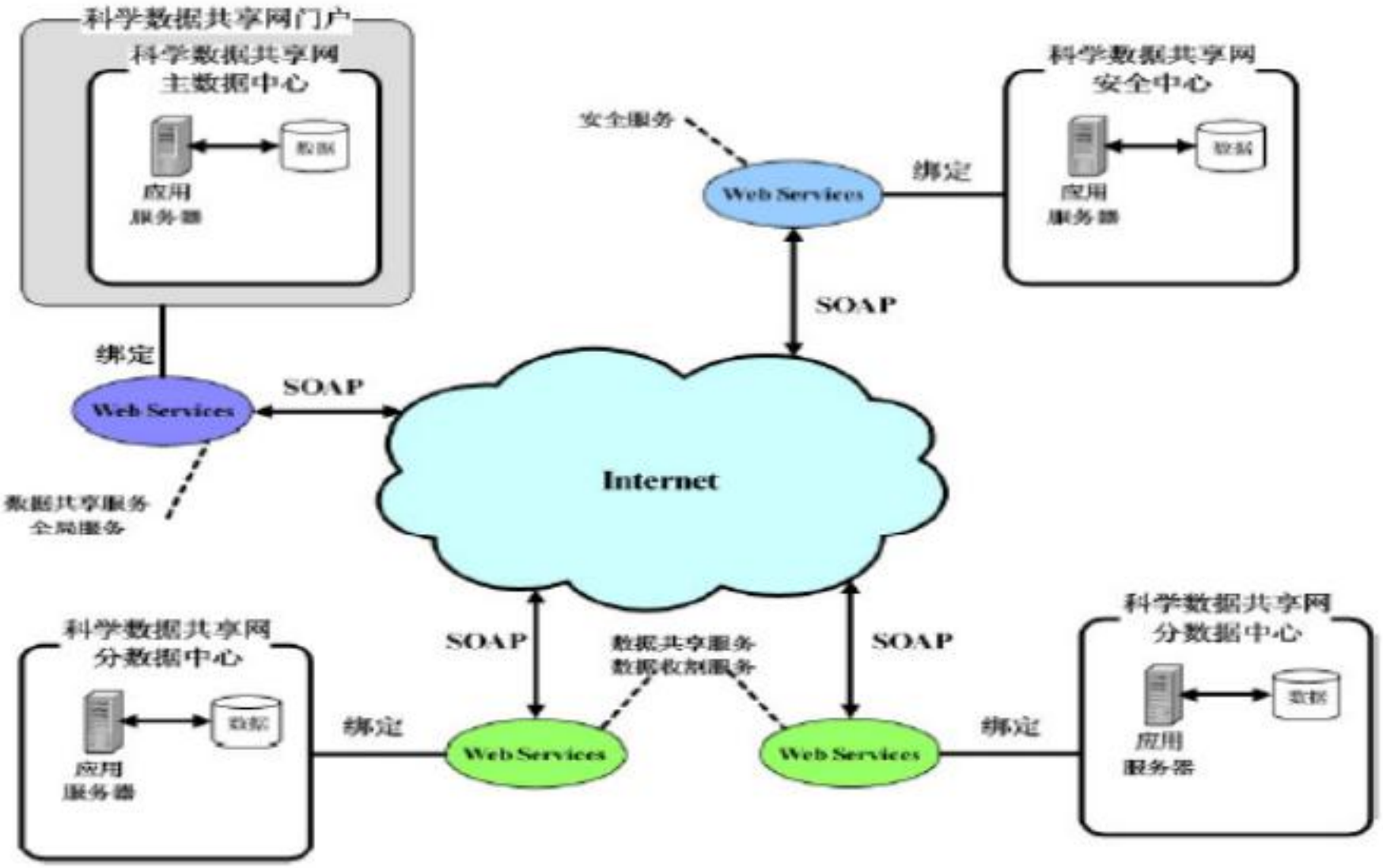
对原型系统的分析

- 所有的数据都由“中心”服务器负责存储，并向用户提供服务。
- 这样的结果是所有的用户请求都由中心服务器来响应。即使内部的四个模块部署到不同的服务器上，“平台数据管理”和两个数据库所承担的运算量也是可观的。
- 考虑到未来的科学数据将会越来越庞大，大量的数据都存储到服务器中，对服务器来讲必然是巨大的负担。而且，数据管理和维护的成本也随着数据量的增加而加大。
- “中心”服务器承载了众多服务，因而其运算量会很繁重；因此为了达到性能方面的要求，对“中心”服务器的要求就会比较高，比如：增加内存容量，CPU数量。这也增加了系统的投入成本。有时，仅仅通过提高服务器的性能是不能够达到性能方面要求的。

对原型系统的分析

- 请求都由“中心”服务器做出响应，一旦它出现了故障，无法提供服务，则存储在系统中的科学数据就无法向外界提供共享服务。
  - 补救办法：增加备份服务器，组成集群
- 当系统升级时，也会对所提供的服务造成影响。
  - 客户要求尽量达到7×24小时服务，平均修复时间不超过2小时。实现客户要求相当难度，成本也高。
- 数据都存储在一个系统内，采取了通过Internet上载或是其它途径（光盘、磁盘等方式）提交科学数据的方式。考虑到地学领域的数据通常是较大的地图，网络提交数据的方式会影响到“中心”服务器的数据吞吐量，降低了系统性能。

重新设计的面向服务的体系结构



体系结构说明

- 面向服务的体系结构
  - 门户（主数据中心），安全中心和分数据中心通过由Web Service构建的数据共享服务、全局服务、安全服务相互连接，组成了科学数据共享网的体系结构。
- 新的体系结构划分了主数据中心、分数据中心和安全中心；三类中心分别有各自基于B/S结构的管理系统，相对独立。
- 遵循面向服务的体系结构思想，为了实现数据的共享服务，各个中心将服务内容封装成Web Service，作为其他中心访问本中心数据的入口，并通过Internet传输数据。
  - SOAP 提供一种简单的、可扩展并且功能丰富的 XML 消息处理框架，用于定义高级别的应用程序协议，从而在分布式异构环境中提供更高的互操作性。

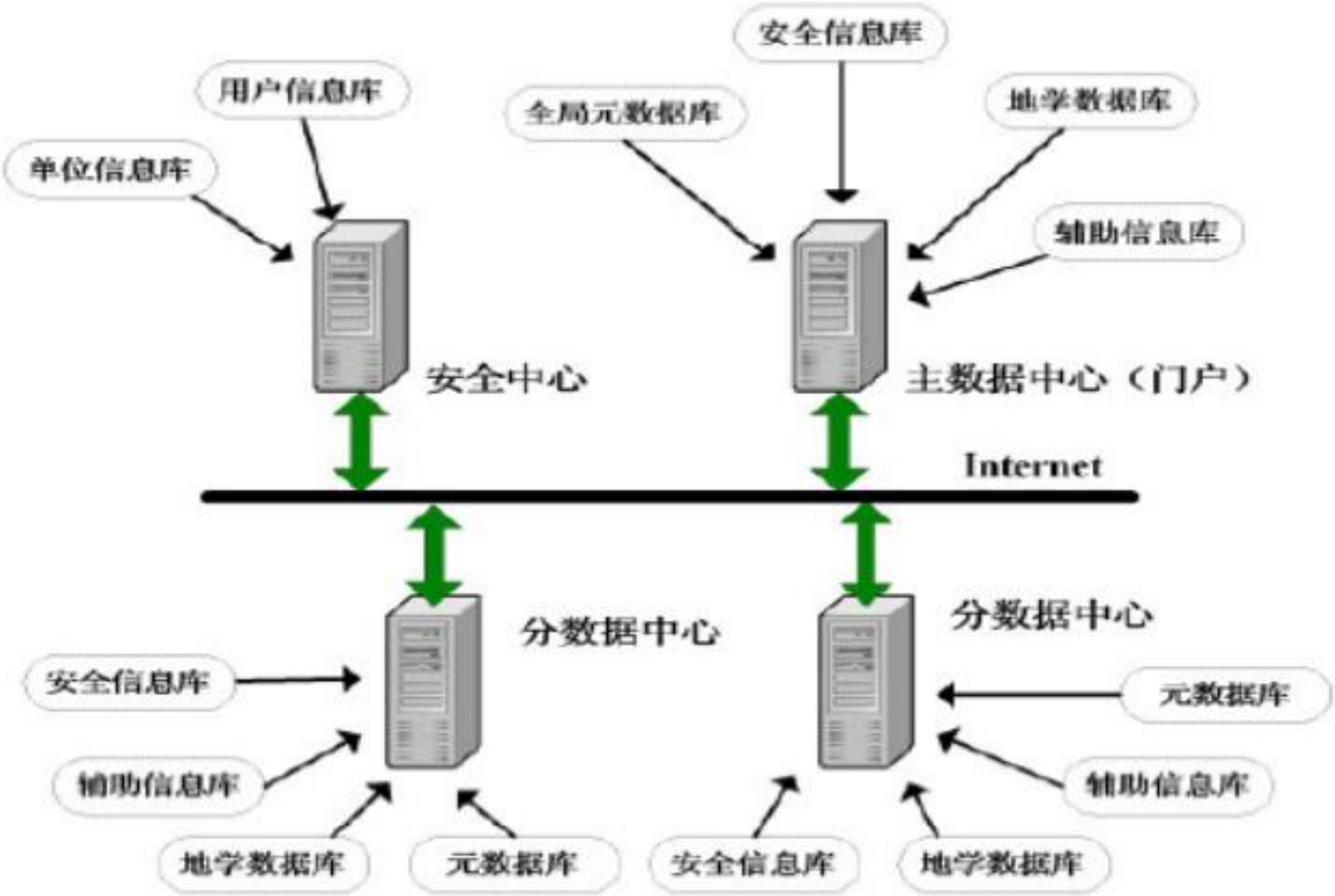
体系结构说明

- 在该系统中，科学数据存储在各个数据中心上；各分数据中心通过数据服务的Web service组件向主数据中心公布其元数据信息，作为实现数据共享的基础（元数据是描述数据的数据）。
- 通过在主数据中心上查找元数据信息，可以快速地获取科学数据的消息信息，定位数据访问入口------某个分数据中心的数据共享服务，然后实现数据的访问。

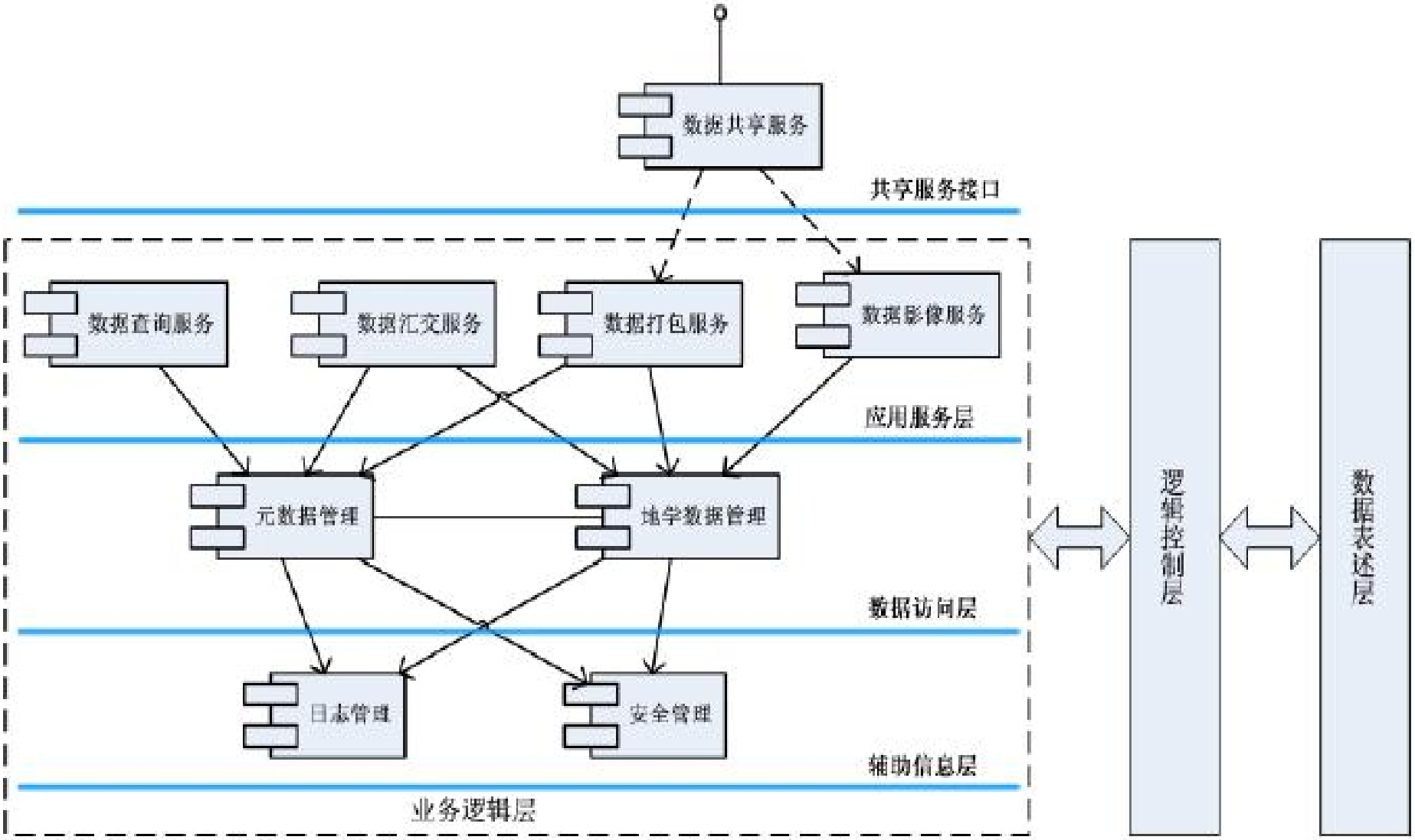
体系结构说明

- 主数据中心作为整个系统共享服务的一个入口，它提供了查询主数据中心上元数据信息的服务；负责向分数据中心转发用户访问科学数据的请求。
- 分数据中心也可以作为共享服务的入口。每个分数据中心都具有各自的管理信息系统，收集和管理某个研究领域内的科学数据，用户可以直接登录某个分数据中心上访问数据。
- 加入了安全中心。用户的基本信息，如密码、住址、所属单位等，都由安全中心保存和维护。安全中心为所有数据中心提供了用户的身份验证、维护的安全服务。
- 用户访问数据的权限则由各个数据中心独立地设置和管理。

各中心的信息存储结构


安全信息库
用户信息库
全局元数据库
地学数据库
单位信息库
辅助信息库
安全中心
主数据中心（门户）
Internet
分数据中心
分数据中心
安全信息库
元数据库
辅助信息库
辅助信息库
地学数据库
元数据库
安全信息库
地学数据库

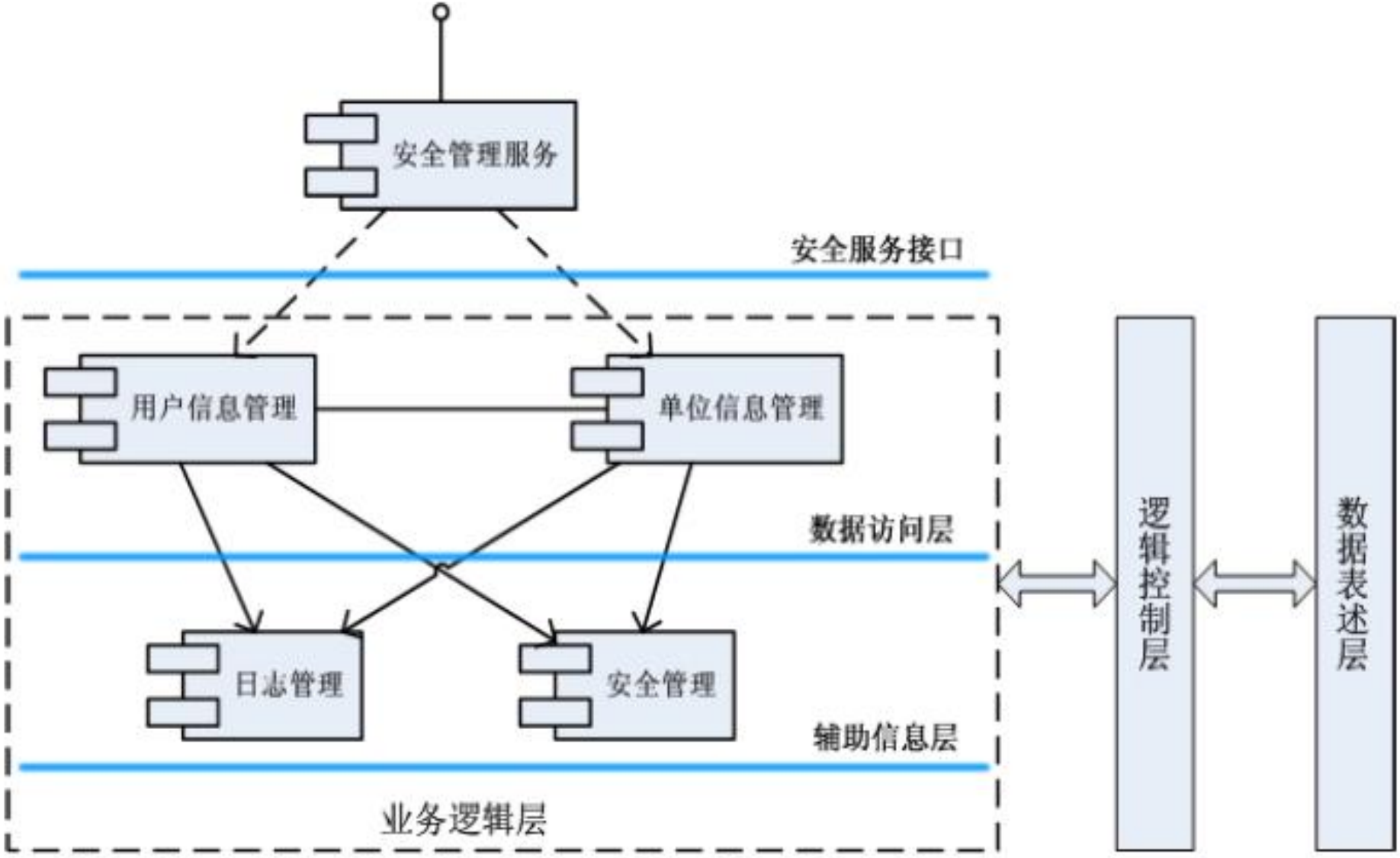
数据中心的分层体系结


数据共享服务
共享服务接口
数据查询服务
数据汇交服务
数据打包服务
数据影像服务
应用服务层
元数据管理
地学数据管理
数据访问层
日志管理
安全管理
辅助信息层
业务逻辑层
逻辑控制层
数据表述层

数据中心的分层体系结构

- 分层体系结构：某一层功能和实现的变化只是上下层有关（低耦合，可扩展、组件复用）
- 安全管理：访问权限
- 日志管理：多种操作的记录
- 数据访问层：审查、发布数据的操作
- 应用服务层：多个共享服务组件
- 共享服务接口：访问接口、入口，重用部分应用服务组件
  - 主数据中心：　全局服务
  - 分服务中心：数据收割（为主数据中心收集元信息）
  - 注意连接件设计

安全中心的体系结构


安全管理服务
安全服务接口
用户信息管理
单位信息管理
数据访问层
日志管理
安全管理
辅助信息层
业务逻辑层
逻辑控制层
数据表述层

体系结构分析

- 面向服务的体系结构是一种松耦合、协议和实现独立的体系结构；松耦合与随需应变是SOA的两大特点。
- 采用这种体系结构，可以重用已有系统作为分数据中心；允许异构数据的存在和访问；并且能够给系统带来良好的可维护性和可扩展性。
- 新的系统中，最主要的一点变化就是：科学数据将不再统一由“中心”服务器存储和管理，改为由数据所有者（科研工作者或是科研院所）负责存储和管理。

数据共享网可维护性解决策略

- 在科学数据共享网系统中，在不同的层次设计中分别采用了“间接化、封装、分离”。
  - 首先，整个系统的分布式体系结构设计，采用了SOA的体系结构设计，这种设计是将数据共享服务封装为Web Service，由特定的Web Service提供数据共享服务。 Web Service的引入，将共享数据内容和共享服务实现细节分离。当实现细节发生变化或是增加新的服务时，对原有系统的运行影响很小。
  - 在数据中心和安全中心的系统设计上，采用了分层的设计。这是一种间接化的体系结构设计策略，上层的共享服务需要通过数据访问层提供的接口访问数据；当底层数据存储结构变化后，对上层的共享服务的影响较小。

数据共享网可维护性解决策略

- 数据中心和安全中心的web应用系统采用的是MVC模式，将模型、控制、视图有机地分离。